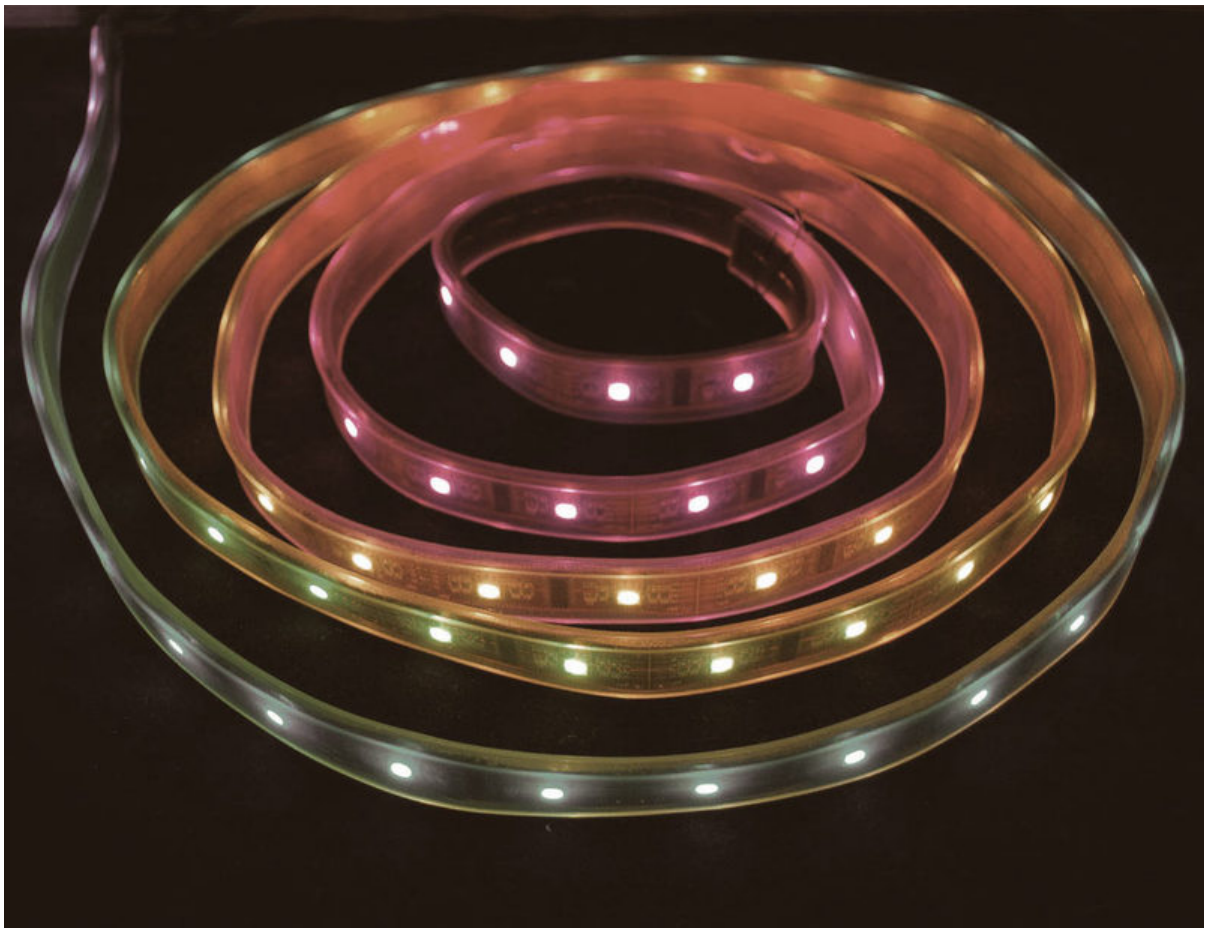
HL1606 LED Strip

Created by Phillip Burgess



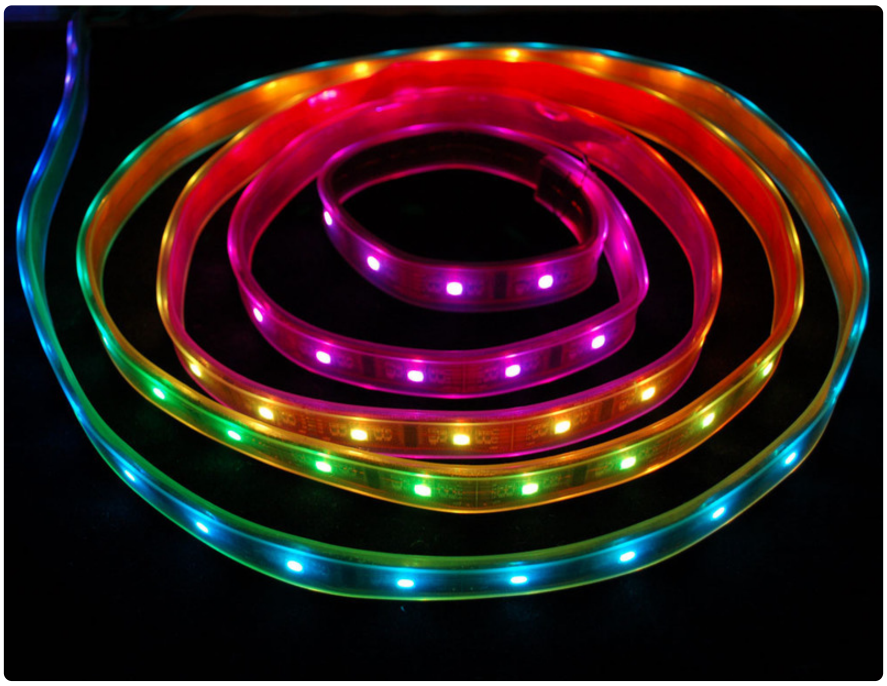
https://learn.adafruit.com/hl1606-led-strip

Last updated on 2024-06-03 01:09:51 PM EDT

Table of Contents

Overview

Please note that Adafruit no longer sells this specific type of LED strip; the tutorial is for the benefit of customers with an existing collection. For new projects we recommend our LPD8806 digital LED strips or 12mm or 36mm LED pixels.



We love some good LED blinking as much as the next person but after years of LED-soldering we need something cooler to get us excited. Sure there are RGB LEDs and those are fun too but what comes after that? Well, we have the answer: **Digital LED Strips**! These are flexible circuit boards with full color LEDs soldered on. They take a lot of LED-wiring-drudgery out of decorating a room, car, bicycle, costume, etc. The ones we carry come with a removable waterproof casing.

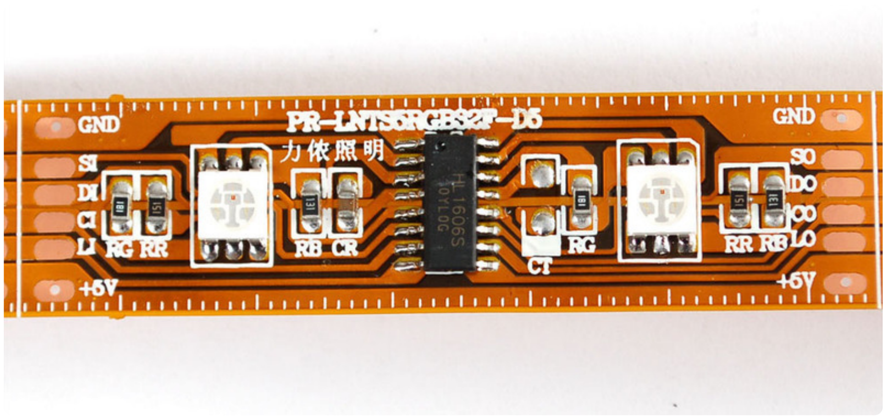
There are two basic kinds of LED strips, the "analog" kind and "digital" kind. Analog-type strips have all the LEDs connected in parallel and so it acts like one huge tri-color LED; you can set the **entire** strip to any color you want, but you can't control the individual LED's colors. They are very very easy to use and fairly inexpensive.

The Digital-type strips work in a different way. They have a chip for each LED, to use the strip you have to send digitally coded data to the chips. However, this means you can control each LED individually! Because of the extra complexity of the chip, they are more expensive.

The strip is made up of 2.5" segments. Each segment is independent and so you can cut the strip down on the segment boundaries, or extend them, or split them up, etc.

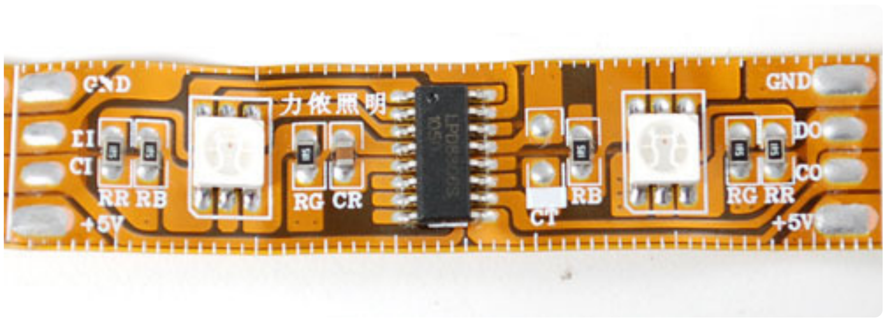
We’ve carried two different types of addressable LED strips. If you ordered before August 2011 then you received the HL1606 type, otherwise you probably have the LPD8806 type. Its not too hard to tell the difference. Look at the segments of the strip and find the 'pads' on the side of each 2.5" segment. The HL1606 strips have six pads on each side. the LPD8806 have four pads on each side.

This is what the HL1606-based strip segments look like:



If you have the HL1606-based strip, this is your tutorial — read on!

The LPD8806-based strip looks like this (its thinner, and has fewer 'pads' on the side):



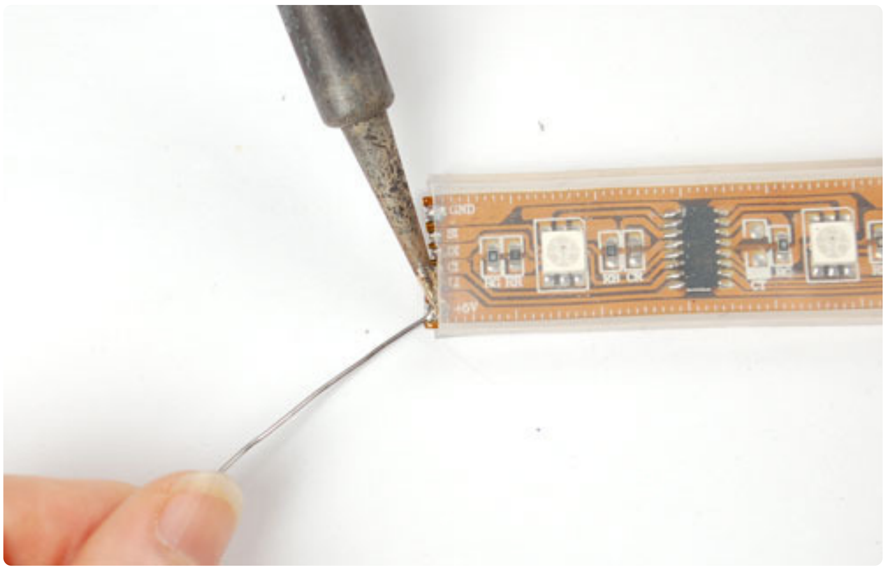
If you have the LPD8806-based strip, that type [now has its own separate tutorial](https://adafru.it/aHG) (https://adafru.it/aHG).

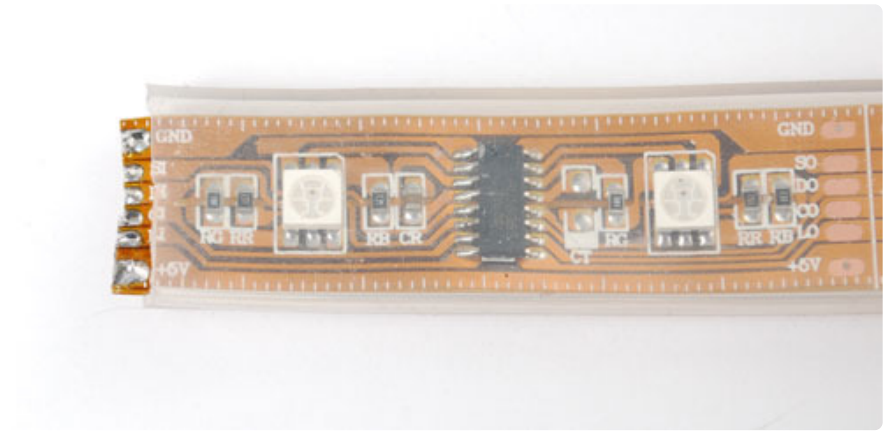
Wiring

The toughest part of the project is probably just soldering to the strip. First, cut the piece to the length you want, on the cuttable boundaries. Next, tin the 6 **INPUT** pins (make sure you're connecting to **SI/CI/DI/LI** which is input, not the corresponding **SO/etc** pins!)

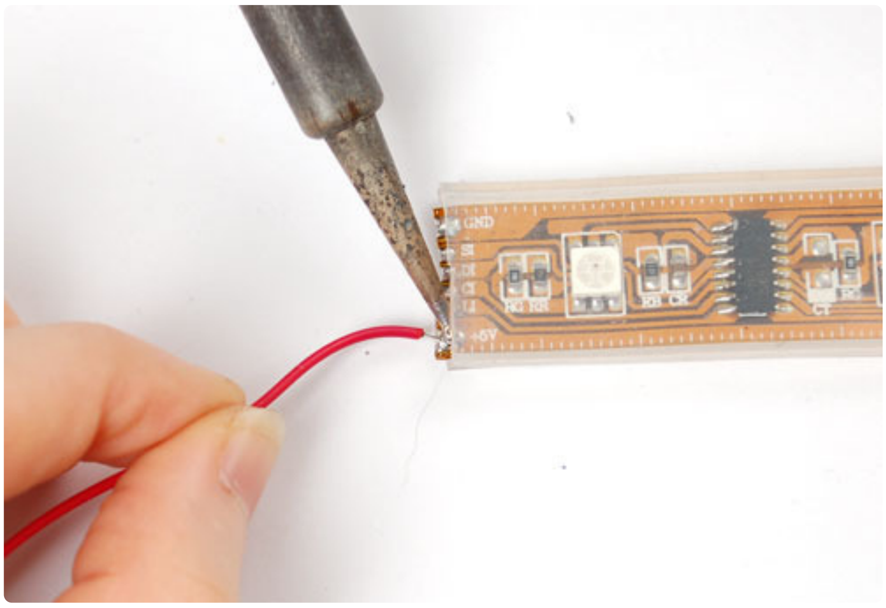
Some strips arrive with wires pre-soldered for testing at the factory, but these might be at either end of the strip and aren't necessarily useful. Check three times to make sure you're connecting to the INPUT side! Just because you have a strip with wires attached does not mean they're on the right end.

Tin the pads by carefully melting a little solder onto the pads:

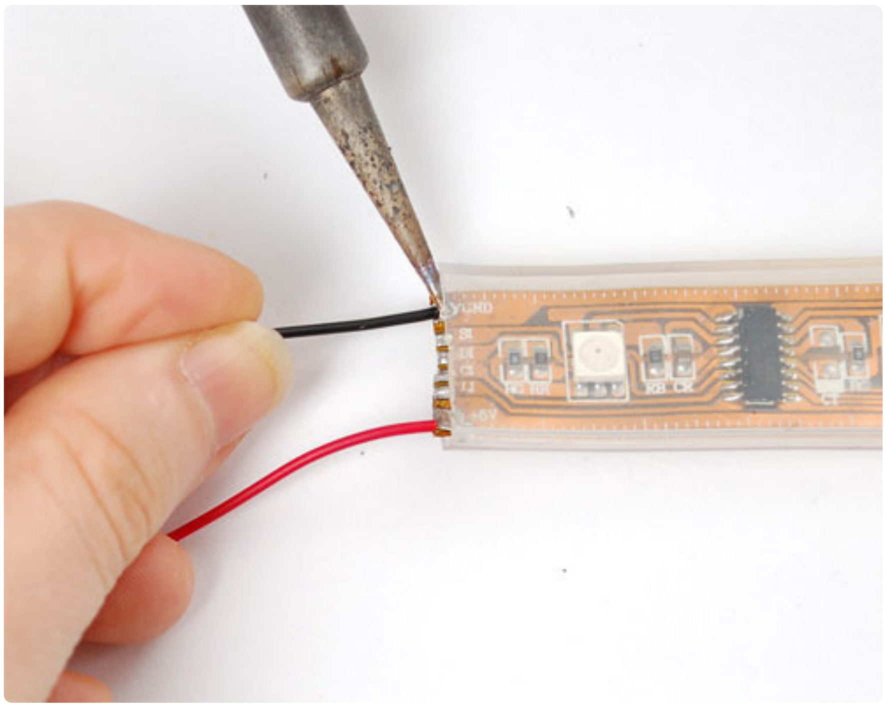


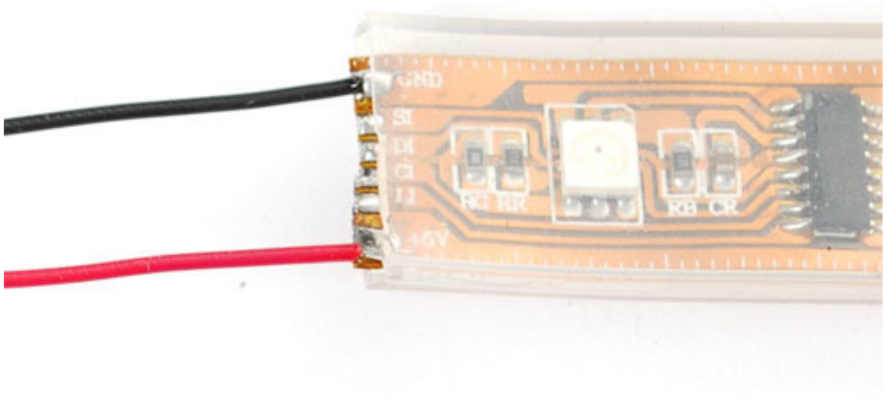


Solder the +5V power wire to the +5V pin. We'll use red:

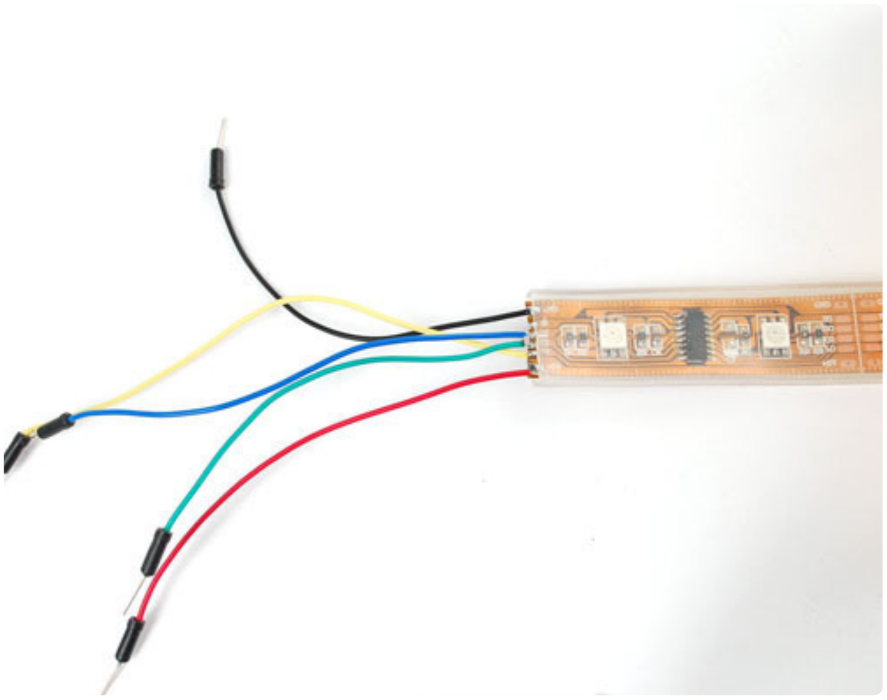


Connect the ground signal/power pin to **GND.** We'll use black:





Follow up by connecting the data lines. We won’t be using the **SI** pin (the strobe function of the chip is not supported by our example code), so connect Yellow to **LI** (la tch), Green to **CI** (clock), and Blue to **DI** (data):



Great! You're now ready to use the strip. You may want to use heatshrink to provide a secure cover for the wires, or stuff hotglue in the end, which will do the same.

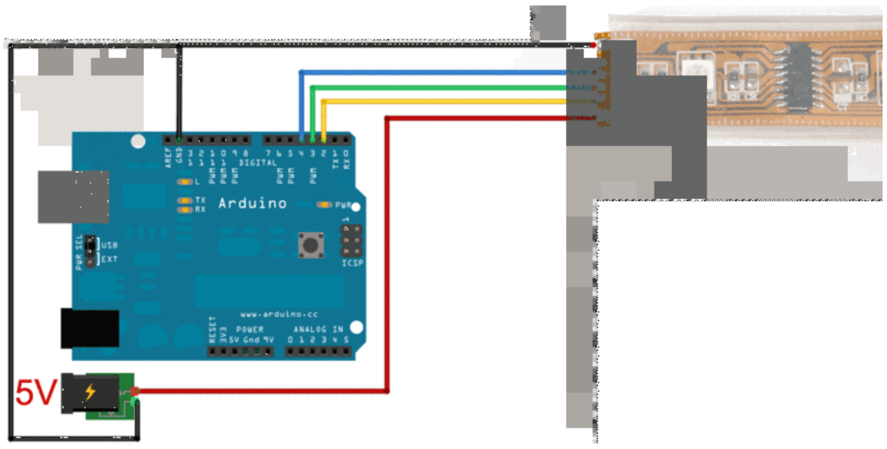
Basic Usage

The HL1606 is not a common chip for most people, so the best way to explain it is to say its basically a 74HC595 shift register. Like a '595 there is an SPI input and then there is a shift-output so you can chain them. The HL1606 has 6 outputs and they're specifically for driving LEDs. The most basic way to use them is to set each LED on or off. This means you can have up to 8 primary 'colors' on an LED: red, yellow, green, teal, blue, violet, white and black.

There are some pros and cons to driving the strips this way...
Pro: Very simple, easy and fast. Use any 3 pins, No interrupts or constant updating required.
Con: Only a handful of colors.

Let's get the strip up and running using this method to start.

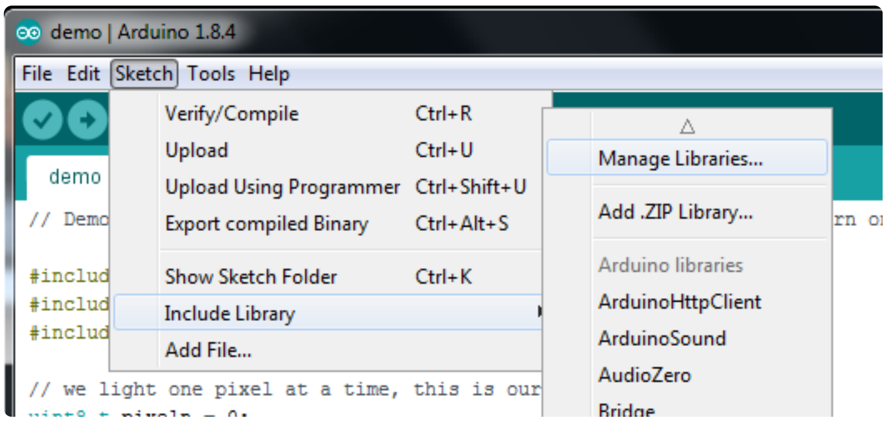
The most important thing to remember is that you need a lot of current (power) to drive these strips, so you will need to arrange a 5V power supply. This test will require about 1 Ampere per meter!



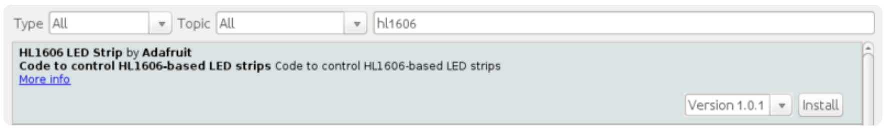
Note in the image above that the 5V can come from a separate power supply that can provide the power you need. Be sure to tie the grounds together and check the polarity...sticking -5V by accident into the strip could be a sad and expensive mistake. We'll be using an Arduino to demonstrate the strip but the code can easily be ported to your favorite microcontroller.

Note that we have **Latch** connected to digital I/O pin #2, **Clock** connected to #3 and **Data** to #4

To install the required library, first, open up the Arduino library manager:



Search for the **HL1606** library and install it



Type All Topic All hl1606

HL1606 LED Strip by **Adafruit**
Code to control HL1606-based LED strips Code to control HL1606-based LED strips
More info

Version 1.0.1 Install

We also have a great tutorial on Arduino library installation at:
http://learn.adafruit.com/adafruit-all-about-arduino-libraries-install-use (https://adafru.it/aYM)

You should now see a new **example** folder called **HL1606strip** and inside, an example called **basicPatterns**. Upload that sketch to your Arduino. You should see the following:

You can change how long the strip is by adjusting the object creation (instantiation) line:

```
HL1606strip strip = HL1606strip(STRIP_D, STRIP_L, STRIP_C, 32);
```

The last argument "32" is the number of LEDs to address. Count how many are in your strip! The display may be wonky otherwise

The **basicPatterns** sketch has many examples of how to set the color of each pixel by calling

```
 strip.setLEDcolor(n, color);
```

where **n** indicates which LED you want to change and **color** is **RED, YELLOW, GREEN, TEAL, BLUE, VIOLET, WHITE,** or **BLACK**. After you've set the pixel color, you need to **write** the changes to the strip by calling

```
 strip.writeStrip();
```

writeStrip() isnt very fast, it will take a few milliseconds to write the changes and it takes longer the more LEDs there are. So change all the LED's you want at once and then write them!

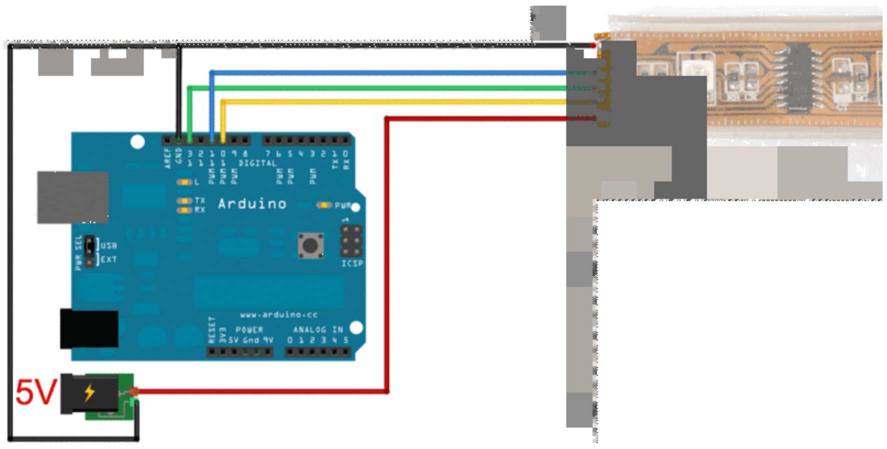
Advanced Usage

Now that you have the basics down, we can get a little more complicated. The really fun part about color LEDs is not just having 8 primary colors but having hundreds or thousands of colors! Again, as we said before, the HL1606 is a rather stupid chip, it is just a shift register. It doesn't really have a PWM system built in which is why it is so low power and low cost. However, we can coax it into display many colors by writing data to the strip really fast. This will PWM the entire strip and will create a blended color effect.

The trade off with the added color-space is that we need to use an interrupt (we use timer #2) to refresh the strip constantly and that the arduino has to do a bunch of crunching in the background.

Pro: Hundreds/thousands of colors!
Con: Uses timer #2, must use hardware SPI on pins 11, and 13, uses background CPU.

To wire this up, we'll have to make a small change. The clock and data lines must now connect to the hardware SPI pins to be fast enough. On atmega168/328 Arduinos, this means 11 and 13 are used for **Data** and **Clock** output (for the Mega, pins 51 and 52). The **latch pin** (L) can be any pin but pin 10 (Arduino) or 53 (Mega) but it **MUST BE AN OUTPUT!**



Now visit our github repository (https://adafru.it/aHJ) and click on the **Download ZIP** button near the top left to download a zip of the library and examples. Uncompress the folder, rename it **HL1606stripPWM** and make sure that inside that folder is the cpp and .h files. Then copy it to your Documents/Arduino/Libraries folder. See our tutorial for more details (https://adafru.it/aYG).

Restart the Arduino software. You should see a new **example** folder called **HL1606stripPWM** and inside, an example called **colorswirl**. Upload that sketch to your Arduino. You should see the following:

(Our camera's sensor didn't film the PWMing very well, its not that flickery in person)

This sketch and library is a little more complex than the one before but should be pretty easy to adapt. Change the object instantiation so the first argument is the number of LEDs in the strip.

```
HL1606stripPWM strip = HL1606stripPWM(32, latchPin);
```

You can then set the color LED resolution, hardware SPI interface speed and how long you're willing to spend on PWMing the strip:

```
  // You can customize/control the pulse width modulation and color
  // resolution by setting how many bits of PWM you want per LED
  // For example, 3 bits is 8 different PWM values per LED and 9 bits, 512
  // values for full color. 4 bits is 16 PWM per LED, 12 bit color with
  // 4096 different colors available.
  // Increasing the PWMbits by 1 means you need *TWICE* as much CPU !!!
  // We suggest starting with 3 and tweaking the other variables to get
  // the fastest SPI and maximum CPU. Then try upping this to 4. For short
  // strips (like 1 meter) that are ok with SPIdiv of 16, you can try 5
  strip.setPWMbits(3);

  // We use the built-in hardware SPI module. We can change the speed
  // of the module to push data out faster. In theory, HL1606's should work with
  // the SPI divider set to 16 but we found that this makes some strips
  // spaz out. Start with 32 and once it works try reducing it to 16
  // If you're lucky, you can even try 8
  // Valid divider values are: 2, 4, 8, 16, 32, 64, and 128, dont try others!
  strip.setSPIdivider(32);

  // all the hard work of running the strip is done in an interrupt
  // we can configure the interrupt so that we spend more or less
  // time running the strip, letting you do other stuff like sensors
  // or an LED or whatever. Set it between 0 and 100, where 100 means
  // higher quality colorstrip display but no time for anything else.
  strip.setCPUmax(70);    // 70% is what we start at
```

The initial settings are a good place to start. You can then tweak the values as necessary. Although it may seem like 70% CPU is a lot, the vast majority of Arduino projects we have seen use only maybe 10% of the CPU usage, a lot of time is spent waiting for input.

Updating the SPI divider to be lower (faster) is 'free' so do that first. Then you can change the PWMbits as you'd like, and finally the CPU max to get good refresh performance.